LEVERAGING MARKUP LANGUAGE DATA FOR SEMANTICALLY LABELING TEXT STRINGS AND DATA AND FOR PROVIDING ACTIONS BASED ON SEMANTICALLY LABELED TEXT STRINGS AND DATA

Field of the Invention

This invention relates to document creation and document viewing program modules. More particularly, this invention relates to methods and systems for leveraging markup language data applied to text strings or data for semantically labeling text strings or data and for providing actions based on semantically labeled text strings or data.

Background of the Invention

Since the advent of the computer and software age, software developers have attempted to provide functionality to users that are contextual in nature. For example, software has been developed for detecting that a user is entering a date into a computer-generated document as the user is typing the date. Before the user can complete the date, the software application pops up today's date to the user and offers to automatically complete the date for the user. Other software has been developed for offering users helpful actions in response to certain data entered by the user. For example, if the user's word processor application recognizes that the user has entered a date, the user may be offered an action that will launch the user's electronic calendar to allow the user to check appointments or to verify information in her calendar for the entered date. It would be helpful to users if such systems could recognize and provide actions for a variety of different types of text strings and data, such as names, dates, stock symbols, book titles, etc. Unfortunately, recognition of such text strings and/or data types may be difficult.

In the case of name recognition, words entered into a text document by a user may be compared against a list of known names to assist in the recognition of a given word as a name. On the other hand, a text string such as a five-digit number that is intended by the user to indicate a zip code may not be readily recognizable and distinguishable from other five-digit numbers included in the text. Accordingly, the five-digit text string entered by the user may not be recognized as a zip code even though the user intends that text string to indicate a zip code.

In recent years, markup languages such as Extensible Markup Language (XML) have been developed to apply structure to text and data where text and data may be tagged with markup language elements to provide contextual structure to the text or data. For example, all person names in a text document may be annotated using Extensible Markup Language (XML) tags to provide structure to the document associated with text or data entered into the document of the type "person name." Subsequently, applications may be created for parsing the document to utilize data annotated with such markup language structure. For example, an application routine may be created to parse a text document and to utilize the XML structure associated with all names contained in the document to extract the names contained in the document for some other use. It would be helpful to leverage markup language annotation of a text or other data document to assist in recognizing text strings or data elements for providing helpful actions on those recognized text strings or data objects.

Because markup languages such as XML allow a wide range of creativity in naming markup language elements to be applied to text or data, it is common for different entities, such as two different companies to use slightly or greatly different XML element naming associated with the same type of text or data. For example, a first company may utilize an XML tag <name> for annotating names in a text or data document, and a second company may utilize an XML tag <personname> for annotating names in a text or data document. The two different entities, for example, companies, may create different sets of actions that may be called on when certain text or data types are recognized in their respective text or data documents. If the two entities decide for some reason to allow utilization of each other's text or data documents, such as the case in contract negotiations or a merger or acquisition between the two entities, or in any case in which one document creator is allowed to use a second document creator's text or data documents, it would be helpful if actions associated with recognized text strings or data objects associated with a first document marked up with XML structure according to a first XML definition used by one entity could be called upon for use in a second document in association with recognized test strings or data marked up with XML elements that are equivalent to differently named XML elements utilized in the first document.

Accordingly, there is a need for a method and system for leveraging markup language structure applied to text strings and data for providing helpful actions based on recognized and labeled text strings and data. It is with respect to these and other considerations that the present invention has been made.

Summary of the Invention

Embodiments of the present invention provide methods and systems for leveraging markup language data applied to text or data for providing helpful actions on certain types of text or data such as names, addresses, dates, stock symbols, book titles, etc.

According to one aspect of the invention, text or data marked up with markup language data, such as Extensible Markup Language (XML) data, is passed by a host application, such as a word processor, spreadsheet application, web browser and the like, to one or more action dynamically linked libraries (DLL) for actions associated with the marked up text or data. For example, actions such as send mail, add to personal contacts, and the like, may be provided to text or data marked up with markup language data associated with a name. For another example, actions such as order this book may be provided for text or data marked up with markup language data associated with a book. After all available actions are identified, the actions are made available for use by the host application in connection with the marked up text or data.

According to another aspect of the invention, after any applicable actions for the marked up text or data are identified, the host application parses a namespace or schema library for markup language data types that have been established as equivalents to markup language data types associated with the text or data. Any equivalent data types are passed back to the action DLLs to determine if any additional actions are available for the marked up text or data based on the equivalent markup language data types. After all available actions are identified, the actions are made available for use by the host application in connection with the recognized and labeled text or data.

According to another aspect of the invention, text or data marked up with markup language data may be passed to one or more recognizer DLLs for leveraging the markup language data to recognize and label text or data not marked up with markup language data so that helpful actions may be applied to the recognized text or data. The recognizer DLLs utilize markup language data associated with the text or data to assist recognition and labeling of text or data. For example, XML markup of a text selection as a name may be used by a recognizer DLL to determine that text annotated with an XML name tag is indeed a name. For another example, XML markup of a city and state text selection may be used by a recognizer DLL to determine that an adjacent unannotated numeric text selection is a zip code. After all unannotated (not marked up with markup language data) text or data is recognized and labeled after one pass to an applicable recognizer DLL, the text selection may be passed back to the recognizer DLL along with associated XML data and any recognition and labeling data established during the previous pass through the recognizer DLL. During the second or subsequent pass through the recognizer DLL, the recognition and labeling previously applied along with associated XML data is utilized by the recognizer DLL to recognize and label text or data not recognized on a previous pass. The recognition process may continue iteratively until no new text or data is recognized.

These and other features and advantages, which characterize the present invention, will be apparent from a reading of the following detailed description and a review of the associated drawings. It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention as claimed.

Brief Description of the Drawings

Fig. 1 is a block diagram showing the architecture of a personal computer that provides an illustrative operating environment for embodiments of the present invention.

Fig. 2 is a block diagram that shows software architecture for recognizing, labeling, and performing actions on strings of text and/or data according to various embodiments of the present invention.

Fig. 3 illustrates a simplified block diagram showing interaction between a host application, a recognizer DLL, an action DLL, and a namespace/schema library for recognizing, labeling, and performing actions on recognized text and/or data according to embodiments of the present invention.

Figs. 4, 5, 6 and 7 illustrate computer screen displays showing illustrative text strings and applicable actions for application to recognize text strings and showing illustrative XML markup of given text strings.

Fig. 8 is a flow diagram illustrating steps performed by a method and system of the present invention for leveraging markup language data applied to text or data for recognizing and labeling text strings or data and for providing helpful actions on recognized text strings or data.

Detailed Description of the Preferred Embodiment

As described briefly above, embodiments of the present invention are directed to methods for leveraging markup language structure and data applied to text or data for providing helpful actions based on recognized and labeled text strings and data. In the following detailed description, references are made to the accompanying drawings that form a part hereof, and in which are shown by way of illustration specific embodiments or examples. These embodiments may be combined, other embodiments may be utilized, and structural changes may be made without departing from the spirit or scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense and the scope of the present invention is defined by the appended claims and their equivalents.

Referring now to the drawings in which like numerals represent like elements throughout the several figures, aspects of the present invention and the exemplary operating environment will be described. Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of program modules that execute in conjunction with an application program that runs on an operating system on a personal computer, those skilled in the art will recognize that the invention may also be implemented in combination with other program modules. Additional aspects of an illustrative operating environment and software architecture for implementing the various embodiments of the present invention are described in U.S. Patent Application No. 09/588,411, entitled "Method and System for Semantically Labeling Strings and Providing Actions Based on Semantically Labeled Strings", which is expressly incorporated herein by reference.

Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Turning now to Fig. 1, an illustrative computer architecture for a personal computer 2 for practicing the various embodiments of the invention will be described. The computer architecture shown in Fig. 1 illustrates a conventional personal computer, including a central processing unit 4 ("CPU"), a system memory 6, including a random access memory 8 ("RAM") and a read-only memory ("ROM") 10, and a system bus 12 that couples the memory to the CPU 4. A basic input/output system containing the basic routines that help to transfer information between elements within the computer, such as during startup, is stored in the ROM 10. The personal computer 2 further includes a mass storage device 14 for storing an operating system 16, application programs, such as the application program 205, and data.

The mass storage device 14 is connected to the CPU 4 through a mass storage controller (not shown) connected to the bus 12. The mass storage device 14 and its associated computer-readable media, provide non-volatile storage for the personal computer 2. Although the description of computer-readable media contained herein refers to a mass storage device, such as a hard disk or CD-ROM drive, it should be appreciated by those skilled in the art that computer-readable media can be any available media that can be accessed by the personal computer 2.

By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EPROM, EEPROM, flash memory or other solid state memory technology, CD-ROM, DVD, or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer.

According to various embodiments of the invention, the personal computer 2 may operate in a networked environment using logical connections to remote computers through a TCP/IP network 18, such as the Internet. The personal computer 2 may connect to the TCP/IP network 18 through a network interface unit 20 connected to the bus 12. It should be appreciated that the network interface unit 20 may also be utilized to connect to other types of networks and remote computer systems. The personal computer 2 may also include an input/output controller 22 for receiving and processing input from a number of devices, including a keyboard or mouse (not shown). Similarly, an input/output controller 22 may provide output to a display screen, a printer, or other type of output device.

As mentioned briefly above, a number of program modules and data files may be stored in the mass storage device 14 and RAM 8 of the personal computer 2, including an operating system 16 suitable for controlling the operation of a networked personal computer, such as the WINDOWS XP operating system from MICROSOFT CORPORATION of Redmond, Washington. The mass storage device 14 and RAM 8 may also store one or more application programs. In particular, the mass storage device 14 and RAM 8 may store an application program 205 for creating and editing an electronic document 24. For instance, the application program 205 may comprise a word processing application program a spreadsheet application, a contact application, and the like. Application programs for creating and editing other types of electronic documents may also be used with the various embodiments of the present invention.

Referring now to Fig. 2, an illustrative software architecture for use in conjunction with the various embodiments of the present invention will be described. The architecture shown in Fig. 2 includes an application program 205, such as a word processor application program, a spreadsheet application program, or other type of application program for creating and editing electronic documents. The application program 205 may also comprise a Web browser.

The application program 205 is able to communicate with a recognizer dynamically linked library ("DLL") 210 and an action DLL 215. As will be described in greater detail below, the recognizer DLL 210 controls one or more recognizer plug-ins 220A-220N and the action DLL 215 controls one or more action plug-ins 225A-225N. According to embodiments of the present invention the recognizer DLL and the action DLL may be a shared software module integrated with the application program 205.

According to one embodiment of the invention, the recognizer plug-ins 220A-220N and the action plug-ins 225A-225N may be automation servers. Automation servers are well-known software components that are assembled into programs or add functionality to existing programs running on operating systems such as the WINDOWS XP operating system from MICROSOFT CORPORATION of Redmond, Washington. Automation servers may be written in a variety of computing languages and can be plugged and unplugged at runtime without having to recompile the host program. The recognizer plug-ins and action plug-ins may also be individual or integrated DLLs accessible by the application program 205.

According to one aspect of the invention, text or data marked up with markup language data, such as Extensible Markup Language (XML) data, is passed by the host application 205, such as a word processor, spreadsheet application, web browser and the like, to one or more action DLLs for actions associated with the marked up text or data. According to embodiments of the present invention, the application 205 annotates the marked up text or data in the document with user interface functionality to allow a user to identify the labeled text or data as a recognized word or data. For example, a name or other marked up text may be annotated with a dotted line underneath the recognized name or other or data item.

The application program 205 may then provide actions that are associated with the recognized string or data. In order to provide actions on the marked up string or data, the application 205 may pass data associated with the recognized string or data to the action DLL 215. The action DLL 215 manages the action plug-ins 225A through 225N that are executed in order to perform the action. The action plug-ins 225A-225N may be packaged with the application program module 205 or may be written by a third party to perform particular actions that are of interest. The action plug-ins 225A-225N provide possible actions to be presented to the user based on the type of word or data passed to the application plug-ins 225A-225N. For example, if data associated with a name is passed to the action DLL 215, an action plug-in 225A that may be responsible for providing electronic mail actions on recognized names may return a list of actions to the application program 205 such as "send e-mail", "add-to e-mail addresses", etc.

The recognizer DLL 210 handles the distribution of text strings and data from an electronic document being edited by the application program 205 to the individual recognizer plug-ins 220A-220N. The recognizer plug-ins 220A-220N recognize particular strings in an electronic document, such as a word processing document or a spreadsheet document. The recognizer plug-ins 220A-220N may be packaged with the application program module 205 or they may be written by third parties to recognize particular strings of interest. Typically, the recognizer DLL 210 passes strings to the recognizer plug-ins 220A-220N in single sentences, paragraphs or cell value increments. However, strings may be passed to the recognizer plug-ins 220A-220N in other sizes and formats.

According to embodiments of the invention, text or data marked up with markup language data may be passed to one or more recognizer DLLs for leveraging the markup language data to recognize and label text or data not marked up with markup language data so that helpful actions may be applied to the recognized text or data. The recognizer DLL 210 in conjunction with a word breaker application breaks the text or data into discrete words and passes the discrete words to one or more recognizer plug-ins 220A-220N for recognition according to each of the individual recognizer plug-ins. Each plug-in may be programmed to recognize a certain category of text or data. For example the recognizer plug-in 220A may be programmed to recognize names, the plug-in 220B may be programmed to recognize addresses, the plug-in 220C may be recognized stock symbols, and so on. For example, if the recognizer plug-in 220A is programmed to recognize names, the plug-in 220A may compare each word received from the recognizer DLL to a database of names such as a user contacts list or telephone directory. A stock symbol recognizer plug-in may compare words passed to it against the database of stock symbols. An address recognizer plug-in may likewise compare text or data against the database of addresses.

A recognizer plug-in may also use more complex logic to recognize and label text or data. For example, an address plug-in may recognize a two-letter state designator as a state, and then the recognizer plug-in may use that recognized text to assist the plug-in in determining that a word immediately preceding the state designator is a city. Likewise, a five-digit number immediately following a state designator may be more readily recognized as a zip code because of its position relative to the state designator.

According to an embodiment of the present invention, information associated with text or data may be passed with the text or data to the recognizer DLL and onto applicable recognizer plug-ins. As will be described in greater detail below, if the text or data is marked up with markup language data, such as Extensible Markup Language, for example, the markup language data associated with a given text string or data may be passed to the recognizer plug-in to assist the plug-ins in labeling text or data. According to another aspect of the invention, text or data may be passed through a given recognizer plug-in iteratively and a given plug-in may utilize text or data recognized from a previous iteration to assist in further recognition during a subsequent iteration.

Referring now to Fig. 3, according to an embodiment of the present invention, a set of equivalence classes may be constructed for one or more semantic labels so that actions provided on a given semantic label may also be provided for an equivalent semantic label. For example, the markup data <name> may be designated as an equivalent to the markup data <personname> and actions constructed for <name> may be mapped to <personname>. Accordingly, if the application 205 passes text marked up with the data <name> to the action DLL, actions constructed for the data <name> and <personname> may be returned by the action DLL for use by the application 205. According to an embodiment of the invention, equivalent actions are obtained from an XML namespace (or schema) library 250. If an equivalent XML namespace and data type are located, any actions associated with the given markup data according to the second XML data type and second XML namespace may be combined with or otherwise used by the application 205 to enhance the actions associated with recognized and labeled text string according to the first XML namespace and first XML data type.

It will be appreciated that the namespace/schema library 250 may be integrated with the host application 205, may be remote from the host application 205 at the users computer for, or may be located remotely from the user computer for accessible by the host application 205 via a distributed computing environment. For further detail on the namespace/schema library 250, see U. S. Patent Application entitled: "System and Method for Providing Namespace Related Information," Serial No.: 10/184190, filed June 27, 2002, and assigned to the same Assignee as the present application and which is incorporated herein by reference as if fully set out herein.

Referring still to Fig. 3, if a given text string, for example " John Doe," is labeled as a name, as described above, but is also marked-up within the electronic document with XML data such that the words "John Doe" are tagged with an XML tag of <employee> according the XML namespace "human resources", that information may be passed to the action DLL 215 and action plug-ins to assist in determining available actions for the recognized text string. When the application 205 references the namespace/schema library 250, a determination is made that the XML tag <employee> of the namespace "human resources" is equivalent to the XML tag <emp> of the namespace "HR" and is equivalent to the XML tag <EES> of the namespace "personnel." For example, one document drafter may have marked-up employee information according to the namespace "human resources", another document drafter may have marked-up a document according to the namespace "HR", and yet another document drafter may have marked-up a document according to the namespace "personnel." According to an embodiment of the invention, the application 205 queries the schema library 250 and resolves mappings between equivalent XML namespaces and data. Once the application has resolved the mappings (meaning it has decided which elements can be treated like other elements) it will then call the action DLL and behave as if the original XML element is actually the element that the to which the action pertains. The action DLL does not need to be aware that the markup data mapping has occurred. For example, if the document being edited by a user has been marked up with XML data according the namespace "human resources", this embodiment of the present invention may allow actions associated with a separate document marked up according to the namespace "hr" to be applied to the first document to enhance the set of actions applicable to a given text string or data, for example the name "John Doe." Operation of this embodiment will be described in further detail below with reference to Figures 7 and 8.

After an action has been chosen from the list of actions, the action DLL 215 manages the appropriate action plug-in 225A-225N and passes the necessary information between the action plug-in and the application program module 205 so that the action plug-in may execute the desired action. Typically, the application program module 205 sends the action DLL 215 an automation request to invoke the action the user has selected.

Figs. 4, 5, 6 and 7 illustrate computer screen displays showing illustrative text strings and applicable actions for application to recognized text strings and showing illustrative XML markup of given text strings. As shown in Fig. 4, an employee payroll document, for example, is being edited by a user. The document includes an employee name, "Joe Smith," and includes other employee payroll data. According to embodiments to the present invention, the words "Joe Smith" are passed to the recognizer DLL 210 and to applicable recognizer plug-ins 220A-220N for recognition and labeling. Because the words "Joe Smith" are returned by the recognizer DLL 210 to the application program 205 as recognized text or data, those words are annotated, for example with underlining, as shown in Fig. 4. Once data associated with the recognized text or data is passed to the action DLL 215 and the applicable action plug-in 225A-225N, actions such as "send mail", "add to contacts", "get employee ID" are returned by the action DLL 215 for use in association with the labeled text or data. According to an embodiment of the present invention, selection of one of the actions applied to the recognized text string or data may launch other required software applications, for example electronic mail applications, electronic contacts applications, and the like to provide the functionality associated with the list of provided actions.

According to an embodiment of the present invention, and as described above with reference to Fig. 3, the employee payroll document illustrated in Fig. 4 is marked-up with XML data according to a first XML namespace or schema, and the actions provided on the recognized name may be particular to the given XML namespace or schema. On the other hand, the example Employee Payroll document illustrated in Fig. 5 may have been created by a second document drafter, for example a document drafter at a separate employee's company, and the second payroll document may have been marked-up according to a different XML namespace or schema. Additionally, a separate set of actions may have been created by the second document drafter to be provided for recognized employee names, as illustrated in Fig. 5.

According to embodiments of the present invention, if a text string, such as "Joe Smith," is recognized as an employee name, XML data associated with the employee name is passed to the action DLL and applicable action plug-ins, as described above with reference to Fig. 3. When the action DLL 215 parses the namespace/schema library 250 to determine whether the XML data applied to the recognized string, for example "Joe Smith" has been established as an equivalent to an employee tag according to a second XML namespace, it may be determined that the XML data for the recognized text string is equivalent to XML data associated with a recognized employee name of the second document illustrated in Fig. 5. Accordingly, the action DLL 215 may utilize these equivalent classes to obtain actions applicable to text strings or data marked-up according to the second namespace/schema, as illustrated in Fig. 5, in order to provide an enhanced list of actions by combining actions applicable to both sets of XML namespaces for use with the recognized text string or data in the first document. As shown in Fig. 6, an enhanced set of actions that includes the actions available to a recognized employee name from both documents are now made available to the recognized name in the first document.

Referring to Fig. 7, illustrative XML markup data is illustrated as annotated to a text selection, for example, a name and address. As shown in the example text selection illustrated in Fig. 7, the name, street, and city have been marked up with XML data, but the state and zip code have not been marked-up with XML data. As briefly described above, when text or data is passed from a given document to the recognizer DLL 210 for recognition and labeling, the recognizer 210 and the associated recognizer plug-ins may utilize markup language data such as the XML data illustrated in Fig. 7 to assist the recognizer DLL 210 and the associated recognizer plug-ins in recognizing and labeling given text strings or data. For example, the inclusion the XML tag <pname> around the words "Joe Smith" may be utilized by the recognizer DLL and the associated recognizer plug-ins to recognized the words "Joe Smith" as a name. Likewise, the annotation of the words "Oklahoma City" with the XML tag <city> may be utilized to assist in the recognition of the words "Oklahoma City" as a city. As should be understood, the text and/or data and XML markup illustrated in Figs. 4, 5, 6, and 7 is for purposes of example only and is not restrictive of the invention as claimed herein.

According to embodiments of the present invention, a given text selection may be processed by the recognizer DLL 210 and the associated recognizer plug-ins 220A-220N to assist in additional recognition and labeling of given text strings or data. For example, if on a first pass to the recognizer DLL 210 and associated recognizer plug-ins, the name, street, and city of the illustrative text shown in Fig. 7 are recognized and labeled, but the string "OK 45678" is not recognized, that text selection may be passed in a second iteration back to the recognizer DLL 210 and associated recognizer plug-ins for a second attempt at recognizing the text strings not recognized during the first pass. During the second pass, the recognizer DLL 210 and associated plug-ins may now leverage the fact that the text string OK is positioned immediately adjacent to the text string "Oklahoma City" that has been labeled as a city in the previous iteration. Accordingly, the recognizer DLL 210 and associated recognizer plug-ins may now determine that there is a high probability that the text string "OK" is a state designator given the two character size of the text string and given its location immediately adjacent to a recognized city.

Likewise, the five-digit number located immediately adjacent to a now recognized state designator may allow the recognizer DLL 210 and the associated plug-ins to label those digits as a zip code either during this iteration through the DLL 210 and associated plug-ins or through a subsequent iteration through the recognizer DLL and associated recognizer plug-ins. According to an embodiment of the present invention, once no additional recognizing or labeling is achieved, the iterative process ends and information associated with the recognized and labeled text strings or data may be passed to the application program 205 for applicable actions, as described above.

Fig. 8 is a flow diagram illustrating steps performed by a method and system of the present invention for leveraging markup language data applied to text or data for recognizing and labeling text strings or data and for providing helpful actions on recognized text strings or data. The method 800 begins at start step 805 and proceeds to step 810 where a text selection is received at the application program 205. It will be appreciated that the application program 205 may be a word processor application, a spreadsheet application, a text editor, a web browser, or other applications capable of receiving text or data entry. At step 812, a determination is made as to whether text or data of the text selection requires recognition and labeling before helpful actions may be applied. If no additional recognition and labeling are required, the method proceeds to step 845, discussed below. For example, if a text selection contains address data where a street and city are annotated with XML markup data, but where text or data representing a state and zip code are not annotated with markup data, the text selection may be passed to a recognizer DLL, as described above, to utilize the annotation of the street and city to assist the recognizer DLL in labeling the state and zip code so that actions associated with states and zip codes may be made available to the application 205. If such additional recognition and labeling are required, then at step 815, the text selection such as a sentence or paragraph and any related XML data applied to the text selection, as described above with reference to Fig. 7, is passed to the recognizer DLL 210 and associated recognizer plug-ins 220A-220N.

At step 820, the recognizer plug-ins 220A-220N perform recognition on the text string passed from the application program 205. For example, if the recognizer plug-in 220A is responsible for recognizing and labeling names, words of the text selection passed to the recognizer plug-in 220A may be compared against a list of known names such as the user's contact list or telephone directory. At step 825, the recognizer DLL 210 and associated recognizer plug-ins utilize any XML data associated with the text string to assist in recognition. For example, referring back to Fig. 7, if the text string "Joe Smith" is passed to the recognizer DLL 210 and associated recognizer plug-ins, the associated XML tag <pname> is also passed to the recognizer DLL 210 and associated recognizer plug-ins to assist in recognizing and labeling the text selection. For another example, if a city is annotated with an XML tag such as <city>, that information may be used by the recognizer DLL and/or associated plug-ins to determine that an adjacently located two-character string is a state.

At step 830, any text strings or data recognized on a first iteration through the recognizer DLL 210 and associated recognizer plug-ins are recognized and labeled accordingly. At step 835, a determination is made as to whether any text strings or data have been recognized in the first iteration through the recognizer DLL 210 and associated recognizer plug-ins. If so, the method proceeds to step 840, and the text or data and associated XML data and labeling information from the last iteration through the recognizer DLL 210 and associated recognizer plug-ins is passed through the recognizer DLL 210 and associated recognizer plug-ins on a second iteration for more recognition. As described above with reference to Fig. 7, on the second iteration through the recognizer DLL 210 and associated recognizer plug-ins, any recognition information obtained on the previous pass through the recognizer DLL and associated recognizer plug-ins is passed to the recognizer DLL and associated plug-ins on the subsequent iteration to further assist in recognition of a given text string or data. For example, as described above with reference to Fig. 7, if the words "Oklahoma City" are recognized as a city on the first pass through the recognizer DLL and associated recognizer plug-ins, that information may be utilized by the recognizer DLL and associated recognizer plug-ins on the second iterative pass to assist in determining that the text string "OK" is a state designator given the two-character size of the text string and given its proximity immediately adjacent to a recognized and labeled city. On a second pass, recognition and labeling of the text string "OK" in association with other recognized and labeled text or data in the text selection may be utilized by the recognizer DLL and associated recognizer plug-ins to assist in the recognition of the five-digit string "45678" as a zip code.

If no additional recognition and labeling are required, or after additional recognition and labeling of text strings or data is performed, as described with reference to steps 815 – 840, the method proceeds to step 845, and the text strings and/or data are passed to the action DLL 215 and associated action plug-ins via the application program 205. According to embodiments of the present invention, any markup language data, for example XML data, applied to the recognized text strings and/or data is passed to the action DLL and associated action plug-ins along with the recognized text strings and/or data to allow the action DLL and/or associated plug-ins to return helpful actions associated with the markup language data or labeled portions of the text or data.

At step 850, the application 205 queries the namespace/schema library 250 for equivalence classes for actionable markup language elements and text or data labels. As should be understood the namespace library 250 may be integrated with the application 205, located in memory separate from the application 205 or located in a remote storage location.

As described above with referenced to Figures 3-7, if a given text string, for example "Joe Smith," illustrated in Fig. 4, is annotated with an XML tag <employee> according to the namespace "human resources," that information may be utilized to determine that the XML tag <employee> of the namespace "human resources," has been established as an equivalent to the class of recognized names annotated with the XML tag <emp> of the namespace "hr." At step 855, if all equivalence classes have been determined, the method proceeds to step 860, and the action DLL and associated action plug-ins build an actions structure for use by the application 205 in providing helpful actions in association with the recognized text string or data. Alternatively, the action DLL 215 and applicable action plug-ins 225A-225N may determine the first set of actions applicable to the recognized text string or data followed by a determination of whether equivalent actions are available. Once all actions for the recognized and or marked up text string or data including actions of equivalent XML namespaces and data types are determined, the set of available actions is passed to the host application 205. At step 870, the host application 205 builds a user interface, as illustrated in Fig. 6, to provide actions on the labeled text or data. The method ends at step 890.

In summary, according to embodiments of the present invention, markup language data, such as Extensible Markup Language data applied to text strings and/or data items may be utilized for providing helpful actions on marked up text or data. Markup language data applied to text or data may also be utilized to assist in the recognition and semantic labeling of a given text string and/or data item. It will be apparent to those skilled in the art that various modifications or variations may be made in the present invention without departing from the scope or spirit of the invention.

Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein.